Extractive Text Summarization Methods Inspired By Reinforcement Learning for Better Generalization

Master Thesis

Yan Virin

Department of Computer Science

Columbia University

yv2163@columbia.edu

May 6, 2019

Abstract

This master thesis opens with a description of several text summarization methods based on machine learning approaches inspired by reinforcement learning. While in many cases Maximum Likelihood Estimation (MLE) approaches work well for text summarization, they tend to suffer from poor generalization. We show that techniques which expose the model to more opportunities to learn from data tend to generalize better and generate summaries with less lead bias. In our experiments we show that out of the box these new models do not perform significantly better than MLE when evaluated using Rouge, however do possess interesting properties which may be used to assemble more sophisticated and better performing summarization systems.

The main theme of the thesis is getting machine learning models to generalize better using ideas from reinforcement learning. We develop a new labeling scheme inspired by Reward Augmented Maximum Likelihood (RAML) methods developed originally for the machine translation task, and discuss how difficult it is to develop models which sample from their own distribution while estimating the gradient e.g. in Minimum Risk Training (MRT) and Reinforcement Learning Policy Gradient methods. We show that RAML can be seen as a compromise between direct optimization of the model towards optimal expected reward using Monte Carlo methods which may fail to converge, and standard MLE methods which fail to explore the entire space of summaries, overfit during training by capturing prominent position features and thus perform poorly on unseen data.

To that end we describe and show results of domain transfer experiments, where we train the model on one dataset and evaluate on another, and position distribution experiments, in which we show how the distribution of positions of our models differ from the distribution in MLE. We also show that our models work better on documents which are less lead biased, while standard MLE models get significantly worse performance on those documents in particular.

Another topic covered in the thesis is Query Focused text summarization, where a search query is used to produce a summary with the query in mind. The summary needs to be relevant to the query, rather than solely contain important information from the document. We use the recently published Squad dataset and adapt it for the Query Focused summarization task. We also train deep learning Query Focused models for summarization and discuss problems associated with that approach. Finally we describe a method to reuse an already trained QA model for the Query Focused text summarization by introducing a reduction of the QA task into the Query Focused text summarization. The source code in python for all the techniques and approaches described in this thesis are available at https://github.com/yanvirin/material.

Contents

1 Introduction

The problem of single document summarization can be described as the following: given a text document the system should produce a short limited in size snippet, which contains the gist of the information and can be easily read and understood by a human. In other words, the produced text snippet should summarize well the content of the document, but also consist of well defined grammatical sentences, one that a human could read and understand.

In abstractive text summarization novel sentences are generated by the summarization system, thus the risk of creating non readable content is elevated. Many of these systems can compress the information pretty well, but lack the ability to create grammatical sentences. In this thesis we will be tackling the problem of summarization using the **extractive** approach, where the summary is constructed by extracting certain sentences from the document, rather creating new ones based on the input. These systems are constrained by the sentences in the input document, as all they can do is decide whether to include or exclude a certain sentence, while breaking sentences into smaller phrases is not allowed. This constraint makes the extractive summarizers worse compressors of information than their abstractive siblings, however also makes them produce perfectly readable output, while many abstractive summarizers tend to struggle with that task.

Classic extractive text summarization systems use different information theory related metrics to score the input sentences and pick the best into the summary until they hit a certain word count limit. The main theme in all of these systems is the computation of diversion of some metric from the mean. The closer to the mean the sentences score, the more appropriate for summary they tend to be marked by the system. Such metrics include KL-divergence, information gain, and other information theory related metrics. These techniques work well in some basic scenarios, however in more specific domains they tend to generate summaries which do not score well with human opinion. Also, they lack the ability to adapt to a particular domain, as the rules on which they rely are fixed and cannot be easily changed.

The question of how to evaluate text summarization is not a trivial one. Intensive research had to be done before researchers could nail down a metric which correlated well with human judgement. That metric was developed by [Lin, 2004] and is called the Rouge metric. Roughly speaking the metric computes the precision and recall of retrieval of unigrams and bigrams in the candidate summary with respect to the golden human abstract of the document. Although intuitively it is not immediately clear this metric should correlate well with real human judgements, in practice it does so surprisingly well. One of the caveats is that the standard implementation of the metric is written in Perl as a standalone script and includes many parameters to tweak. That makes it not really scalable and applicable in many scenarios were large amounts of computations of the metric needs to be executed. In addition to that, for better precision one can exclude stopwords from being counted, and include stemming to improve recall, thus when comparing two different systems it is import to state which parameters were used exactly. This makes Rouge not the perfect candidate metric for evaluation, but currently state-of-the-art non the less. In extractive text summarization, as the output summary is of limited size, precision portion of the metric is not really relevant and researchers focus on the recall component. In abstractive text summarization both parts are relevant, thus the F measure is reported in most cases.

In this thesis we explore several problems with current text summarization systems, which in our opinion prevent them to perform well in practice.

The first problem is the **generalization** problem. It is well known that machine learning based text summarization systems do not perform well on unseen data, partially due to lack of good representative labeled data, and also due to the inadequate learning mechanisms applied to the problem. In this thesis we try to attack both issues, by proposing a new dataset based on news from social media, and by exploring new learning schemes which tend to generalize better on unseen data due to their better space exploration using reinforcement learning inspired methods. The dataset we have built consists of summaries of news written by humans and published on the wikinews website. The content is being reviewed by moderators constantly and we use only approved summaries. These summaries contain more approachable language and thus can be used for learning summarization systems to be deployed in environments where informal language is in use.

The second problem is the **bias to the lead** problem that most news summarization systems tend to have. Due to the fast pacing modern nature of news consumption, editors and writers tend to create content which is biased to the lead, i.e. where the first several sentences of each document contain the most important information, and the rest of the document gives the details. If all content in the world had that property, we would not need to develop models for the summarization task at all, as a system which selects a few first sentences would work perfectly. However that's not the case, and we do need summarization systems which go beyond selecting top portion of the document to get good content summarization. Having that said, most of the available datasets for summarization are heavily lead-biased, as they come from the news realm. When we train machine learning models on such datasets they tend to strongly capture the **positional** features. Strictly speaking, during prediction these models tend to select sentences from the top of the document, which is obviously not the desired behavior. We have made an attempt to create a less lead biased dataset than the existing ones, however our wikinews dataset is not perfect as we will see in the experiments section. We have also designed our learning algorithms to explore the search space in a better way, so that they do not capture positional features as strongly as standard maximum likelihood models do.

Training models with our methods is not a simple task. We use two different types of optimization techniques to train summarization models. One is using sampling from the model itself, and one is sampling from the labels space. Both techniques has their challenges. When sampling from the model, the variance of the estimate of the gradient tends to get very high and different tricks have to be applied to control it and make optimization finish in reasonable time. When sampling from the labels space, a large amount of candidates has to be scored with Rouge, which may take considerable resources. Thus in both cases we have to resort to approximation, which allows us to retain some of the original power of the optimization technique, but also generate models in reasonable time.

Another theme this thesis touches upon is query focused summarization, a problem which arrises in practice and for which standard summarization systems do not provide good answer. We explore usage of Question Answering datasets for training Query Focused summarization systems and show there is a lot in common between the two tasks. We then reduce the problem of Query Focused summarization to QA by showing that the latter system can be used as a proxy for the former one.

2 Research Focus

This thesis has four main sections and is organized as follows:

- First comes the topic of application of loss functions inspired by reinforcement learning for extractive summarization in comparison to traditional maximum likelihood training. We describe related work, different loss functions used for text summarization and some experiments with deep learning models in section 4.

- Then we explore the problem of generalization in text summarization and present our hypothesis as to why machine learning models do not generalize well in section 5. To that end we discuss the labeling techniques applied to create datasets. We also describe our experiments results which show that alternative learning methods may create better generalizable models.

- Further we talk about how sequence models trained with maximum likelihood tend to produce models biased to the lead portion of the document, and strongly depend on the positional features. We describe the problem in section 6 and show that our methods perform better in regards to that problem.

- Finally we claim that in the presence of a search query standard summarization systems do not produce relevant summaries. We define the problem in section 7 and describe how we used the Squad dataset and a QA system to address that problem.

3 Datasets for Text Summarization

Throughout the thesis we are using several datasets for text summarization commonly used by others for reporting results. We also developed a new dataset from Wikinews, which we describe in more details in the next few sections.

3.1 Published datasets

CNN-DM

This dataset was introduced in [Hermann et al., 2015] for the task of reading comprehension, although also can be used for text summarization. Researchers create two datasets by using online newspaper articles and their respective summaries. They have collected around 93k articles from the CNN1 and 220k articles from the Daily Mail2 websites. Both news providers add upon their articles a number of bullet points, summarizing different portions of the information contained in the articles. When we run our experiments on this corpora, we combine both datasets into one and get what we call CNN-Daily Mail dataset, or CNN-DM for short.

DUC-SDS

DUC stands for Document Understanding Conference. DUC proceedings serve as a platform for the dissemination of technical papers written by participants in the DUC workshops, who use the DUC datasets, which were provided each conference year. These datasets are manually curated and thus small, but serve as standard for text summarization tasks. SDS sands for single document summarization. This dataset contains single documents as inputs and golden summaries, which were created by humans as supervision signal.

DUC-MDS

This is yet another type of a dataset from DUC. MDS stands for multi-document summarization. Although summarizers presented in this thesis concentrate on single document summarization task, we do explore multi-document summarization by creating a single document summarization dataset from the DUC-MDS. These DUC datasets tend to be small, and thus we combine datasets from 2001 and 2002 years for training and 2004 for testing. Multi-document datasets contain document sets instead of single documents, which need to be summarized into one summary. In the DUC-MDS dataset documents in one document set contain similar information, taken from different sources but describe same or related events. We combine all the documents in each doc set into one super-document and run our single document algorithms on the that data to produce summaries.

XSUM

Another recently published dataset that we use in this thesis is the XSUM dataset. It was published by [Narayan et al., 2018a] and in that work researchers introduce a new single document summarization task called **extreme summarization**. The idea was to create a very short, one-sentence news summary answering the question “What is described in the article?”. They collect a large dataset for this task by extracting online articles from the British Broadcasting Corporation (BBC). This particular dataset imposes additional difficulties on the extractive summarizers as there is only one sentence they need to match in the human abstract.

3.2 Wikinews Dataset

We made an attempt to create a more versatile dataset based on informal language and potentially lesser lead-bias. To do that we developed a codebase to build a news dataset from the online wikinews service. This section described the details of that portion of the work.

Our dataset is of news articles and their summaries, created by humans and published on the Wikinews repository. Wikinews selectively publishes human summaries of news from all around the world and we decided to use that resource in order to carefully curate a high quality dataset for summarization. The news articles on Wikinews are public and are available for download using the wiki api’s. We decided to use an existing implementation of a client which uses those api’s and targets Wikinews in particular, out of all other Wiki-based services. The package is called pywikibot and the code is implemented in python. We use the package to query for the news articles through HTTP and store the raw results on disk. Unfortunately for the users of this client, the output is formatted in a very custom fashion and we had to write our own parser to be able to correctly parse each article with its sources. The format resembles the regular Wiki data format, but does not have clear start and end markers for each articles and has several exceptions that we had to treat heuristically. Moreover, we had to clean, normalize and filter the data. We got rid of various Wiki related tags in the text, normalized the data by removing punctuation and lowercasing the words, and also ran a naive bayes based sentence segmenter called Splitta(https://github.com/lukeorland/splitta) to split text to sentences.

The recent advances in machine learning made it easy in practice to use word and sentence embeddings as features for different tasks in NLP. Approaches range from very complex sentence embeddings as shown in [Jiao et al., 2018], where authors actually train embeddings on whole sentences, to approaches where sentence embeddings are constructed from more popular pre-trained word embeddings, as in SIF system described in [Arora et al., 2017]

We used the latter simpler approach to create sentence embeddings, as it seems like they do perform well on many NLP related tasks, according to the authors and are very modular, in the sense that you don't need to have a separate apparatus to create the embeddings - the sentence embeddings are just a function of the word embeddings contained in the sentence. Recent development in the area of word embeddings which produced the popular sets Word2vec and Glove vectors, make it very easy to quickly construct the embeddings in practice. Another reason for using a simple combination based approach is that otherwise sentence embeddings might be difficult to analyze and assess their quality. When the embedding is just a weighted linear combination of the word vectors in the sentences, as in our approach, it is very easy to check which words dominate the embedding and then tweak the needed parameters.

One thing that we skip in the SIF implementation is the removal of the first few principal components, a technique which according to the authors removes some of the influence of very common words. We do not argue that this technique is of no value by all means, but according to our analysis in our domain this operation can be skipped, allowing us to more freely apply the embeddings, without any global dependency like the one that PCA enforces (one does not have to submit a large set of sentences in one batch to the embeddings computation code, as no global dependency exists, because there is no need in computing PCA on the entire batch/corpus). To test our sentence embeddings we made the following analysis: for each sentence in each of the Wikinews articles we picked the closest sentence using the cosine similarity metric on the sentence embeddings vectors. Here is one example from that analysis: *"reports say that the plane sent a distress signal at 2:00 am just after it took off from cameroon but it is not known what triggered the distress signal"* was very close to *"a distress signal was later sent out by the plane kenyan airways chief executive titus naikuni said adding that this would have been sent out automatically"*

4 Loss Functions for Summarization

Most of recent research in the area of machine learning for extractive summarization has been focusing more on different algorithms and feature extraction techniques, and less on loss functions and ways to label data. In this thesis we describe two new promising training procedures which incorporate new loss functions. These loss functions require new optimization techniques and a new way of labeling data, departing from the traditional greedy approach. The training procedures are called Minimum Risk Training (MRT) and Rewards Augmented Maximum Likelihood (RAML). These approaches are not novel by themselves and both were applied successfully to neural machine translation before, however as far as we know have not yet been applied to extractive text summarization.

4.1 Related Work

In recent years researchers began to use supervised machine learning techniques to create more sophisticated summarization systems which learn from supervised data and use different feature representations. The sentences are featurized and a loss function with respect to the labels is minimized to produce a set of optimal weights. These approaches tend to differ in the way the sentences are featurized, but for the most part use the same techniques for labeling and similar loss functions to approximate the divergence from the optimal Rouge score computed against a set of golden summaries curated by humans.

As early as in 1995 researchers already were experimenting with supervised machine learning approaches for extractive text summarization. The problem was posed as a classification task where the decision of wether to include or excelude each sentence is made by a classifier. Models like Naive Bayes were applied by [Kupiec et al., 1995] and [Miles, 2002], maximum entropy by [Miles, 2002] and support vectors machines by [Tsutomu et al., 2002]. It is interesting to note that in addition to different general features, which became almost standard in the NLP community since then (e.g. n-grams, tokens weighted by TF-IDF, lexical features like phrases from gazeteers, etc.) researchers used the positional features extensively. Thus we know positional features were strongly used in early research on summarization and it is unlikely that the problem of lead-bias was introduced only recently with the advance of deep learning sequence models. One of the main research focuses of this thesis is how pervasive can be position features.

In addition to representing the problem as an independent classifier, significant body of work has been done in the area of structured predictions for extractive text summarization. Instead of thinking independently about each sentence, the summarization problem can be represented as a sequence problem, similarly to how entity extraction problem is dealt with in modern NLP. Instead of testing each sentence independently, we could reason about the whole sequence of sentences included and excluded from the summary and learn parameters of a model in order to maximize the likelihood of these sequences which rise from our labeled data. Models like the HMM, CRF and structured SVM were all applied to extractive summarization in the past (see [Shen et al., 2007], [Conroy and O'Leary, 2001], [Berg-Kirkpatrick et al., 2011] for reference).

The survey in [Dipanjan and Andre, 2007] describes a slew of approaches starting from HMM's and Log-Linear models to Deep learning architectures all used for extractive single document text summarization.

Since deep learning models began to become the "bread and butter" in recent NLP research in general, text summarization arena has also began to be infiltrated by the use of deep architectures.

Convolutional and Recurrent neural networks have been recently applied to the problem of extractive text summarization and yielded state-of-the-art results, even so that structured inference in these cases tends to be computationally intractable and simpler inference techniques have to be applied instead. Deep architectures used for text summarization recently include simple multi-layer multi-label classifiers, simple RNN and CNN models, and also encoder-decoder sequence to sequence models. The variety of different architectures generally does not align with loss functions variation, i.e. most of these models use some type of cross-entropy loss. The labels are generated using a simple greedy technique based on local Rouge optimization. In the next sections we introduce two different loss functions, which do explore the space of labels in a better way.

Recently research also includes deep reinforcement learning methods application to extractive text summarization. For example [Yao et al., 2018] present a Deep Q-Network for extractive text summarization. Deep Q-Networks are deep neural networks which directly approximate the Q-values of state and action and learn the induced policy to maximize the long term rewards. Each sentence selection is defined as an action in the RL methodology, and the Rouge score between a candidate summary and the gold human abstract is used for computing rewards.

Another example of direct application of reinforcement learning for extractive text summarization is shown by [Narayan et al., 2018b], where REINFORCE style algorithm is used to directly optimize the policy. This work in particular is similar to our Minimial Risk Training approach, where the risk is minimized with respect to a special Q distribution, constructed over a finite set of samples of candidate summaries.

Recent research has also seen works like [Keneshloo et al., 2018] where domain transfer for text summarization is being solved through reinforcement learning. In that paper the issue of poor generalization of state-of-the-art models on unseen datasets is discussed. They propose a learning framework which uses self-critic policy gradient and show good results on a variety of datasets. They show an interesting ability of the framework to fine-tune models using only a few training samples. This aligns with us developing models for better generalization, where we show the advantages of reinforcement learning ideas applied for extractive text summarization.

4.2 Maximum Likelihood

Labeling in general lies in the basis of all modern supervised machine learning approaches. It is fair to say that any system produced by a supervised machine learning approach can be only as good as the labels which have been fed into the learning algorithm. Labels, along with the features, are used to produce the loss function, which is the objective all supervised methods are supposed to optimize. If the labels are wrong or suboptimal, the loss function no more correctly represents the problem at hand and thus stops being a good objective for optimization. Without a good loss function supervised machine learning does not work and thus it stops working without good labels as well.

Noisy labels is a well known problem in the supervised branch of the machine learning field. As in most cases it is a very expensive and labor intensive process, in practice there is a tendency to use various heuristics to produce cheap labels, even with a set back to the performance down the road. For example, if we want to classify email to spam vs. non-spam using a supervised machine learning approach, we need someone to categorize our emails to these two categories in order to produce a supervised training set. In that context labeling can be seen as only a moderate problem, because it is pretty simple for a human to decide wether an email is a spam or not just from one glance at the content. However it still costs money and takes significant amount of time, as we do need to pay someone to go over all emails in our training set, potentially containing thousands of instances. Instead of doing that we could use a heuristic, according to which we create labels from some implicit signal rather than conscious email reading.

For example, we could examine user behavior if we have access to such information. If users spend time reading a particular email and do not delete it right away, then we could conclude it is unlikely this email is spam. On the other hand, if the users delete the email immediately even without opening it, then it is a good sign they encountered an unwanted content. Is that a perfect approach to get labels? Not at all. There might be different reasons for users not to delete a spam email immediately, for example due to procrastination. But it might be a good enough approximation after all.

When it comes to extractive summarization it seems like it is even a bigger problem to directly get labels from humans. If we wanted to do that, we had to ask people to go over all sentences in each document in the training set and select the ones they think should be in the summary. It would be problematic to reuse these labels for other extractive summarization systems which need summaries with a different length. Moreover, those labels would be totally worthless for abstractive summarization, as they just define the best sentences to extract, and not the optimal summary which could be constructed from the content of the document. Due to the fact that creating such special labeled datasets solely for extractive summarization would be very expensive and it would most certainly produce non reusable outcome, researchers use heuristics to approximate extraction of optimal sentences.

The common procedure most recent extractive summarization systems implement for labeling data includes humans who are being asked to produce a summary for each document in the dataset as free written text, perhaps sometimes using phrases from the original document and sometimes coming up with entirely new content. By minimizing the constrains on how humans perform summarization, we hope to get the most natural summary of the text, so to speak a summary that is truly produced due to human creativity and ability to compress information. This kind of a summary potentially can be used for different related summarization tasks, like abstractive and extractive summarization. However, it does impose a difficulty in coming up with direct labels in the extractive summarization task, finding the indices of the sentences which should be included in the summary by the summarization system. That second step, in which we produce a ”labeled” datapoint out of a document and a human gold summary is a fuzzy process and cannot be well defined universally. How should we decide which sentences are the most suitable to be in the summary, based on a summary written by a human after reading the original content, which might contain new unseen words and phrases?

One way to think about it would be that we want somehow to score each possible summary against the human gold reference and obviously pick the best one. Intuitively, this approach does make sense, however we quickly realize we have a combinatorial problem at hand and that going over all possible summaries and scoring them with some complex metric will be most likely intractable. It does although depend on the complexity of the metric. If the metric is linear with respect to summation, in the sense that the sum of the metric applied to each sentence equals to the metric applied on all the sentences in the summary combined into one piece of text, then we could use a greedy algorithm to optimize it and get the best sentence indices as labels. The optimal metric however has to be correlated to how humans perceive quality of a summary, meaning the higher the value of the metric, the better humans think this excerpt summarizes the document. One such metric was found recently and named Rouge - it is based on recall and precision in the token space between the proposed system summary and a golden human reference. Multiple experiments have shown this metric indeed correlates well with the perceived quality and thus became a standard in the field of summarization, similar the BLUE score in the domain of machine translation.

However the Rouge metric is not linear with respect to the sum. The sum of the Rouge R1 of each individual sentence does not equal to the Rouge of the concatenation of all the sentences. The main reason for that is the ability of Rouge to penalize duplication, a known problem in summarization systems. Adding a sentence with content already seen in the candidate summary does contribute to the metric, even if this content is really relevant with respect to the reference. This forces summarization systems to select new content with each additional sentence added to the summary. It also creates a dependency between the first sentence added to the candidate summary and the following one, making it impossible to decompose the optimization and use a greedy algorithm to extract all the labels with an optimal outcome. It seems also intractable to compute the Rouge metric for all possible combinations of sentences in a given document. That means we have to use some heuristic in order to approximately search for the best combination of sentences, and that leads us to the greedy approach that seems to work in most cases:

- For each document, we create the candidate summary, initially an empty set of sentences further to be marked with the label 1.
- We look for a sentence which maximizes Rouge with respect to the reference summary provided by humans for that document.

- We include that sentence into the candidate summary, and continue the process, by selecting the next sentence which maximizes Rouge (not including the selected one) of the candidate summary with the human abstract: we try to add each sentence to the candidate summary and look for one which makes the candidate summary to achieve the best score.

- Note that the second computation of Rouge should be done for the candidate summary as a whole, meaning the combination of two sentences together - the first sentence which was chosen and the second one which is being tested at current step. This emphasizes the non linearity with respect to the sum as we explained earlier Rouge has due to the importance of penalizing duplication of content.

- We continue to the third iteration, by adding each sentence (excluding the two selected into the candidate) to the summary, one at a time, computing the Rouge and selecting the one sentence which maximizes the score. We stop the process when we hit the length limit of the summary to be extracted (we truncate the last sentence if needed)

In figure 1 we show schematically how the labels are selected according to the above described process.

The process remains greedy in the sense that the order of adding sentences might be significant for Rouge computation and does not guarantee the optimal combination of sentences to be selected. In some cases this standard approach of extracting greedy labels may not work optimally. Our hypothesis is that when data is noisy or there is some duplication of important information in the document, the above described approach would fail to label all possible good summaries, as it is geared to select only one which is the best. For example, let's say there is a sentence which includes the words ”talking to a stranger might be dangerous”, and suppose this sentence was selected to be in the summary. Another sentence like ”strangers might be dangerous in certain situations” might not get selected into the summary, as it contains duplicated information. If a sentence is not selected into a summary, then that particular sentence gets the 0 label, and when we use Maximum Likelihood to train a model which optimizes cross entropy of the labels, we might get a suboptimal

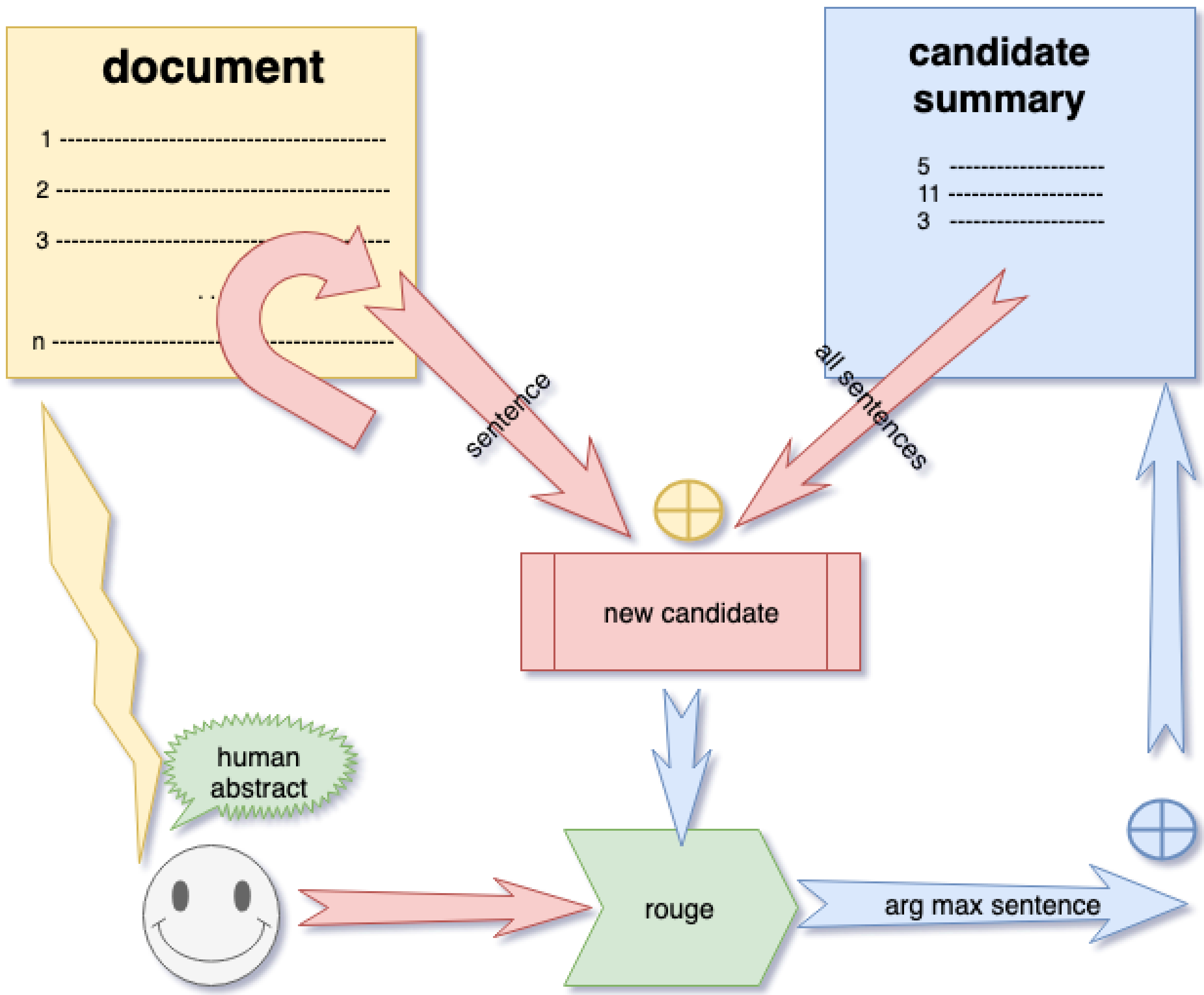


Figure 1: The scheme for the labels selection algorithm. In the begining the **candidate summary** is empty. We start going over each sentence in the document and add it to the new candidate to get it scored. The yellow plus sign represents concatenation of the sentences to produce a new candidate, and the blue plus sing represents the addition of the best selected sentence to the candidate.

performance overall, as two sentences with similar information have two different labels and may confuse the learning process.

Maximum Likelihood (MLE) is a popular approach for machine learning based extractive text summarization. Let us define the problem of summarization as a maximum likelihood problem: We will assume we have a labeled training dataset with S documents, and each document s having $N(s)$ total sentences. The label vector for each document s is defined as $y^{(s)} \in \{0,1\}^{N(s)}$. The featurized document is represented by a vector $x^{(s)}$, and a conditional probability distribution is defined over the labels given the documents, and parameterized by vector θ.

$$\hat{\theta}_{MLE} = arg\max_{\theta}\left\{\mathcal{L}(\theta)\right\} \tag{1}$$

where

$$\mathcal{L}(\theta) = \sum_{s=1}^{S} \log P(y^{(s)}|x^{(s)};\theta) \tag{2}$$

$$= \sum_{s=1}^{S}\sum_{n=1}^{N(s)} \log P(y_n^{(s)}|x^{(s)}, y_{<n}^{(s)};\theta) \tag{3}$$

with S being the set of documents, $N(s)$ the number of sentences in each document, and $y^{(s)}$ being the label of the sentence s in the domain of $\{0,1\}^{N(s)}$

The partial derivative with respect to a model parameters θ_i is given by:

$$\frac{\partial\mathcal{L}(\theta)}{\partial\theta_i} = \sum_{s=1}^{S}\sum_{n=1}^{N(s)} \frac{\partial P(y_n^{(s)}|x^{(s)}, y_{<n}^{(s)};\theta)/\partial\theta_i}{P(y_n^{(s)}|x^{(s)}, y_{<n}^{(s)};\theta)} \tag{4}$$

There has been a growing body of evidence that this approach might be suffering from a particular problem described in details in [Ranzato et al., 2016]. MLE usually uses cross-entropy loss focusing on single sentence inclusion errors to maximize the probability of the next correct label of the sentence, which might not always correlate well with corpus-level evaluation metrics such as Rouge. Another known issue related to MLE training on sequences is the **exposure bias**. Architectures which during decoding condition on previous output suffer from this problem, as they are not exposed to their own predictions distribution, but rather to the real data distribution, present in the training data.

For example in machine translation the previous word is frequently fed into the decoder as input, while during test time the decoder has only its own predicted word in previous step to use as input for the next prediction. Feeding its own prediction during training introduces too much variance and makes training impossible or at least very time consuming in practice. On the other hand using gold words during training introduces the exposure bias issue, which might impose a real problem during prediction and cause the model to introduce errors. In our architecture the decoder acts independently for each of the sentences and thus does not suffer from this problem.

Given all the above, we think it is important to introduce new training procedures and loss functions which capture the nature of the problem in a better, more holistic way. As we can see the gradient in equation 4 is fairly simple to optimize directly, if the probability distribution P is represented as a differentiable function, e.g. with a deep neural network. We will see in the next few sections that this is not always the case with other types of loss functions. We will eventually propose a compromise, in which the loss function is both simple to optimize and does explore the space better than MLE, due to the ability to directly optimize for Rouge.

4.3 Minimum Risk Training

Minimum Risk Training (MRT) has been applied in many fields as a technique to minimize an expected loss and thus get optimal behavior of a model in expectation. This method allows for direct optimization of the metric of interest instead of using approximate labels. The basic idea of MRT is to use the evaluation metric directly to build a parameterized differentiable risk function and optimize that objective to find best parameters. Most metrics like Rouge actually has to be maximized, thus a negation of the metric should be used in loss function construction. We have implemented this approach for extractive text summarization by creating a differentiable loss function which follows the negative expected sum of rouge scores with respect to the human curated reference summaries for each document in the training set. Training works by taking samples from the model and using these to approximate the gradient. This approach has been used in classical machine translation (SMT) ([Koehn et al., 2003], [Smith and Eisner, 2006], [He and Deng, 2012]) and neural machine translation ([Shen et al., 2015]).

We will use $\mathcal{L}(y, y^{(s)}) = -Rouge(sum(y), sum(y^{(s)}))$ to represent the discrepancy between the model prediction and the gold best outcome, i.e. the human abstract summary of the document. We apply Rouge metric on the summary generated by the system and induced by the labels $y^{(s)}$

which is $sum(y^{(s)})$ and the golden human abstract summary which we denote by $sum(y)$. Notice that $\mathcal{L}$ is based on the Rouge evaluation metric and is not parameterized nor differentiable. We further construct a differentiable risk function based on a negatively smoothed Rouge metric. We can state that MRT is aimed at minimizing the following risk which is the expected loss with respect to the posterior distribution:

$$\mathcal{R}(\theta) = \sum_{s=1}^{S} E_{y|x^{(s)};\theta}\left[\mathcal{L}(y, y^{(s)})\right] \tag{5}$$

$$= \sum_{s=1}^{S} \sum_{y \in \mathcal{Y}(s)} \mathcal{P}(y|x^{(s)};\theta)\mathcal{L}(y, y^{(s)}) \tag{6}$$

where $\mathcal{Y}(s)$ is the set of all possible summaries for document s.

The training objective of MRT is to minimize the risk on the training data:

$$\widehat{\theta} = arg\min_{\theta}\left\{\mathcal{R}(\theta)\right\} \tag{7}$$

Intuitively, while MLE is maximizing the likelihood of the gold label sequence, MRT training objective is trying to discriminate between candidate summaries with high and low Rouge. In order to be able to optimize with gradient descent we need to derive the gradient of the risk shown above. To that end we will have to use a well known **log** trick, where we represent the derivative of an intractable sum as an expectation and estimate that quantity with Monte Carlo sampling, similarly how estimation of complex integrals works. We begin by taking the derivative of the risk:

$$\frac{\partial \mathcal{R}(\theta)}{\partial \theta} = \sum_{s=1}^{S} \sum_{y \in \mathcal{Y}(s)} \mathcal{L}(y, y^{(s)}) \cdot \partial \mathcal{P}(y|x^{(s)};\theta)/\partial \theta \tag{8}$$

Then we acknowledge that the following holds true due to **derivative of log** formula:

$$\partial \mathcal{P}(y|x^{(s)};\theta)/\partial \theta = \mathcal{P}(y|x^{(s)};\theta) \cdot \partial \log \mathcal{P}(y|x^{(s)};\theta)/\partial \theta \tag{9}$$

By plugging that in to the derivative of the risk we get that the risk can be expressed as an expectation, as we got the $P(\cdot)$ out due to the log trick above:

$$\sum_{s=1}^{S} \sum_{y \in \mathcal{Y}(s)} \mathcal{L}(y, y^{(s)}) \cdot \mathcal{P}(y|x^{(s)}; \theta) \partial \log \mathcal{P}(y|x^{(s)}; \theta) / \partial \theta \tag{10}$$

$$\begin{aligned} &= \sum_{s=1}^{S} E_{y \in \mathcal{Y}(s)} \left[\mathcal{L}(y, y^{(s)}) \cdot \partial \log \mathcal{P}(y|x^{(s)}; \theta) / \partial \theta \right] \\ &\cong \sum_{s=1}^{S} \frac{1}{n} \sum_{y=1}^{n} \mathcal{L}(y, y^{(s)}) \partial \log \mathcal{P}(y|x^{(s)}; \theta) / \partial \theta \end{aligned} \tag{11}$$

Using that trick we get the following derivation over all the training data:

$$\frac{\partial \mathcal{R}(\theta)}{\partial \theta} = \sum_{s=1}^{S} E_{x|y^{(s)};\theta} \left[\mathcal{L}(y, y^{(s)}) \times \textstyle\sum_{n=1}^{N(s)} \frac{\partial P(y_n^{(s)}|x^{(s)}, y_{<n}^{(s)}; \theta)/\partial \theta_i}{P(y_n^{(s)}|x^{(s)}, y_{<n}^{(s)}; \theta)} \right] \tag{12}$$

As we see from the above equation, we do not need to differentiate the loss function itself thus any loss function can be used. Another advantage is that MRT can be applied to any end to end neural machine learning architecture. However, the main difficulty with MRT lies in the computation of the expectation which has to be calculated over all possible candidates which is clearly impractical. That's why we follow [Shen et al., 2015] and define the risk over a subset of the space by introducing an approximate distribution Q:

$$\tilde{\mathcal{R}}(\theta) = \sum_{s=1}^{S} E_{y|x^{(s)};\theta,\alpha} \left[\mathcal{L}(y, y^{(s)}) \right] \tag{13}$$

We define $\widehat{Y}(s)$ to be a subset of $Y(s)$ which is the set of all possible summaries, i.e. all combinations of sets of t sentences from document s. We get $\widehat{Y}(s)$ by sampling from the model distribution (in our experiments we used $|Y(s)| = 100$ and $t = 3$)

$$= \sum_{s=1}^{S} \sum_{y \in \widehat{Y}(s)} \mathcal{Q}(y|x^{(s)}; \theta, \alpha) \mathcal{L}(y, y^{(s)}) \tag{14}$$

where $\widehat{Y}(s) \subset Y(s)$ and $\mathcal{Q}$ defined over $\widehat{Y}(s)$:

$$\mathcal{Q}(y|x^{(s)};\theta,\alpha) = \frac{\mathcal{P}(y|x^{(s)};\theta)^{\alpha}}{\sum_{y' \in \widehat{Y}(s)} \mathcal{P}(y'|x^{(s)};\theta)^{\alpha}} \tag{15}$$

The hyper-parameter $\alpha \in [0, 1]$ controls the manner in which distribution Q is approximating theoretical distribution of Rouge over the entire space of summaries $Y(s)$. With small values of α the Q distribution becomes flatter smoothing away sharp differences between Rouge values of different summaries, while with larger values the distribution become spiky, and in the limit concentrates around the larger values of Rouge (in our experiments we have seen that small values around 0.001 work best).

And finally we differentiate with respect to the parameters θ to get the derivative of $\tilde{\mathcal{R}}$:

$$\frac{\partial \tilde{\mathcal{R}}(\theta)}{\partial \theta} = \alpha \sum_{s=1}^{S} E_{y|x;\theta,\alpha} \begin{bmatrix} \partial \mathcal{P}(y|x^{(s)};\theta)/\partial\theta \times \mathcal{P}(y|x^{(s)};\theta)^{-1} \times \\ (\mathcal{L}(y, y^{(s)}) - E_{y|x;\theta,\alpha}[\mathcal{L}(y, y^{(s)})]) \end{bmatrix} \tag{16}$$

This approach has a lot in common with the REINFORCE based algorithms introduced in [Williams, 1992], where gradient of expected reward with respect to the policy parameters is used during optimization of the policy directly, instead of going through the value function in a reinforcement learning setting. Furthermore [Sutton et al., 2000] proved that a version of policy iteration with arbitrary differentiable function approximation is convergent to a locally optimal policy. The relation between direct policy optimization in reinforcement learning and MRT training approach can be seen through the long-term reward function $\rho(\pi)$ with respect to a policy π and its gradient, when compared to the risk equations shown above:

$$\begin{aligned} \rho(\pi) &= \lim_{n\to\infty} \frac{1}{n} E[r1 + r2 + r3 + ... + r_n|\pi] \\ &= \sum_s d^{\pi}(s) \sum_a \pi(s,a) R(s,a) \end{aligned} \tag{17}$$

One practical issue we faced when implementing MRT for extractive text summarization was related to computation of Rouge for each sample of documents. As original implementation of Rouge was done in perl, we could not efficiently incorporate it inside our pytorch implementation of the MRT deep learning model. We had to implement a simplified version of Rouge, which does not always follow the original metric, but is closely related. It computes the overlap between the tokens found in the summary and the tokens found in the golden human reference. As the original metric includes special stopwords handling, stemming, and multiple other options which are commonly used, our risk minimization problem was posed as an approximation to the original one and definitely made the task of finding good set of parameters more difficult.

Our implementation of MRT is written in pytorch and uses categorial sampling to sample from the model to compute the risk. The pseudo code for this procedure is given in the following algorithm:

Algorithm 1 MRT risk computation; σ designated softmax operation, and S($\cdot$) the sigmoid function

```
1: procedure ComputeRisk(logits)
2:     $probs \leftarrow S(logits)$
3:     $samples \leftarrow categorical(probs)$
4:     $\log \mathcal{L}(samples) \leftarrow \log(samples)$
5:     $\mathcal{Q} \leftarrow \sigma(\log \mathcal{L}(samples) \cdot \alpha)$
6:     $\mathcal{R} \leftarrow SmoothedNegRouge(samples)$
       return $\mathcal{R} \cdot \mathcal{Q}$
```

As we can see in the ComputeRisk function above, it gets as inputs the logits from any differentiable computation graph. It then uses the logits to produce probabilities, and uses those to sample from the model using categorical sampling. A log likelihood is created over the samples and Q values are computed. Then Q values are multiplied by R values, which represent the smoothed negative Rouge scores, to produce the mean expected risk.

4.4 Reward Augmented Maximum Likelihood

We have adapted the Reward Augmented Maximum Likelihood (RAML) approach introduced in [Norouzi et al., 2016] to the extractive text summarization task. At the core of this approach lies the ability to sample from the exponentiated payoff distribution, which allows the approximation of the expected maximum likelihood. As the task of text summarization is highly structured, and it is not trivial to sample from the distribution of all possible summaries, we turned to an approach which approximates importance sampling for this distribution with a special preprocessing algorithm. We go over all possible summaries of certain length (i.e. 3 sentences long), and score all of them with respect to the gold human reference summary for each document. Then we keep only a small number of best scoring summaries and use those to construct the loss function.

In this section we will define $\mathcal{L}_{RAML}$ and $\mathcal{L}_{RL}$, which are the RAML and Reinforcement Learning (RL) loss functions respectfully. We will also see in this section that the MRT approach is very similar to RL and thus suffers from the same variance in gradient estimation problems. We will show that the main reason optimizing $\mathcal{L}_{RAML}$ is easier than $\mathcal{L}_{RL}$ in practice, is that in reinforcement learning the sampling is done from a dynamic distribution, and thus the variance of the estimate is very high, while in RAML we sample from a stationary distribution. There has been a lot of work done in the reinforcement learning community to reduce the variance of that estimate, including actor-critic methods in [Sutton and Andrew, 2017] and [Degris et al., 2012].

In the original approach in [Norouzi et al., 2016] the researches use importance sampling in order to approximately sample from the distribution according to the value of the metric of interest. They use edit distance between the sampled sentence and the gold translated sentence, as they apply the approach to machine translation. In our implementation, we explode the space of labels and go over the candidates to score all of them with Rouge. In optimization time we construct the weighted average MLE loss with respect to the Rouge scores of the top scored candidate summaries. Thus in our approach we do not sample, but rather apply deterministic optimization with a more complex loss function, which incorporates more information coming from multiple possible summaries.

RAML can be seen as a simple and computationally efficient approach to use task reward within a maximum likelihood framework. To make connection between Maximum Likelihood Estimation (MLE), Reinforcement Learning (RL) and Reward Augmented Maximum Likelihood (RAML), we examine their corresponding loss functions:

For MLE we have negative log likelihood:

$$\mathcal{L}_{MLE}(\theta) = \sum_{s=1}^{S} -\log P(y^{(s)}|x^{(s)};\theta) \tag{18}$$

When we minimize this objective, the conditional probability of the targets increases, however at the same time the conditional probability of all alternative outputs decreases without discrimination. According to this approach, all wrong labels are equally wrong, and none of them is preferred over the others, while in reality it is clearly an incorrect approach to follow.

RL is taking a different approach, by looking at all possible labels and trying to minimize the negative regularized expected reward (e.g. Rouge score):

$$\mathcal{L}_{RL}(\theta,\tau) = \sum_{s=1}^{S} \left\{ -\tau\mathcal{H}(p(y|x;\theta)) - \textstyle\sum_{y\in\mathcal{Y}} P(y|x;\theta)r(y,\hat{y}) \right\} \tag{19}$$

where τ is the parameter controlling regularization, $\mathcal{H}$ is the entropy and $r(y,\hat{y})$ denotes the reward, like the Rouge score between summary given by a particular set of labels and the gold human abstract.

If we look back at the MRT equation 6, we see that it looks very similar to equation 19 defined above. Both approaches sample from the dynamic model distribution and thus suffer from the high variance at estimating the gradient.

RAML can be seen as a hybrid between MLE and RL, and the following distribution, called the exponentiated payoff distribution, is central to linking between the two:

$$q(y|\hat{y};\tau) = \frac{1}{Z(\hat{y};\tau)} \exp\left\{ r(y,\hat{y})/\tau \right\} \tag{20}$$

where $Z(\hat{y};\tau)$ is the partition function.

We define then the RAML loss as the following:

$$\mathcal{L}_{RAML}(\theta;\tau) = \sum_{s=1}^{S} \left\{ -\textstyle\sum_{y\in\mathcal{Y}} q(y|\hat{y};\tau)\log P(y|x;\theta) \right\} \tag{21}$$

To see the connection between $\mathcal{L}_{RL}$, $\mathcal{L}_{MLE}$ and $\mathcal{L}_{RAML}$ it is possible to rewrite those using KL-Divergence according to [Norouzi et al., 2016] into:

$$\mathcal{L}_{MLE}(\theta) = \sum_{y \in \mathcal{Y}} \mathcal{D}_{KL}\left(\delta(y|\hat{y}) \| P(y|x;\theta)\right) \tag{22}$$

$$\mathcal{L}_{RL}(\theta,\tau) \cdot 1/\tau + constant = \sum_{y \in \mathcal{Y}} \mathcal{D}_{KL}\left(P(y|x;\theta) \| q(y|x;\tau)\right) \tag{23}$$

$$\mathcal{L}_{RAML}(\theta;\tau) + constant = \sum_{y \in \mathcal{Y}} \mathcal{D}_{KL}\left(q(y|x;\tau) \| P(y|x;\theta)\right) \tag{24}$$

For optimization we need to define the gradients:

$$\partial\mathcal{L}_{RAML}(\theta;\tau)/\partial\theta = E_{q(y|\hat{y};\tau)}\left[-\partial \log P(y|x;\theta)/\partial\theta\right] \tag{25}$$

while:

$$\partial\mathcal{L}_{RL}(\theta;\tau)/\partial\theta = E_{P(y|x;\theta)}\left[-\partial \log P(y|x;\theta)/\partial\theta \cdot r(y,\hat{y})\right] \tag{26}$$

Although RAML is sampling from a stationary distribution, that distribution is tricky to define. It has to be defined over all possible objects in the predictions space and be proportional to the reward associated with that object. In machine translation for example, that distribution would be over all possible translations of a sentence and proportional to the BLUE score, while in extractive summarization it would be over all possible subsets of sentences in a document and proportional to the Rouge score.

As the space in machine translation is merely infinite and it is tricky to actually sample proportionally to the BLUE score of each translation, the approach taken in [Norouzi et al., 2016] is to use stratified sampling and compute edit distance as a proxy to the reward: first a distance is selected randomly and then an output is sampled with this particular distance. We think introducing sampling during gradient computation is destabilizing the process of training and adds variance to the estimate of the gradient. Also, it is tricky to sample from the space of all summaries proportionally to the Rouge score with respect to the golden human abstract. We decided to skip the sampling process alltogether, and instead pre-compute the Rouge scores for all possible summaries of a particular size k(we experimented with k=3 and k=4). This resulted in $\binom{n}{k}$ candidates for each document to compute the Rouge score for. Thus we were not able to use the standard Rouge perl based script as our framework is written in python, and had to define our own Rouge scorer. This Rouge implementation basically calculates the Rouge R1 by looking at how many tokens from the golden summary were also present in the candidate. The pseudo code for the RougeScorer is presented in the following figure:

Algorithm 2 The rouge scorer; $candidate$ and $abstract$ are dictionaries which hold the counts of tokens for the candidate summary and the counts of tokens for the human abstract respectfully,

```
1: procedure RougeScorer(candidate, abstract)
2:     $total \leftarrow 0$
3:     $covered \leftarrow 0$
4:     for key in abstract do
5:         $goldCount \leftarrow abstract[key]$
6:         $total \leftarrow total + goldCount$
7:         $count \leftarrow candidate[key]$
8:         $covered \leftarrow covered + \min count, goldCount$
       return $\frac{covered}{total}$
```

As described above, instead of sampling we use the top scored candidates t (in experiments we use $t = 100$) to compute the weighted cross entropy loss for document D and its corresponding abstract A, in the following way:

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{t} -w_i \cdot \sum_{j=1}^{s} y_j^i \log \hat{y}_j \tag{27}$$

where $\hat{y} \in \{0,1\}^{N(s)}$ is a vector of size $N(s)$ and represents the predictions vector of sentences in document s which should be extracted as a summary, $y^i \in \{0,1\}^{N(s)}$ represents the labels of candidate i, and $w \in R^t$ stands for the weights vector proportional to the Rouge scores $w_i = \frac{RougeR1(candidate_i(D),A)}{\sum_j RougeR1(candidate_j(D),A)}$

4.5 Experiments

The architecture we used throughout all our experiments was a sequence to sequence encoder-decoder with LSTM units, where sentences are encoded by averaging pre-trained word embeddings, and the decoder outputs one probability for each possible sentence in the document independently using a softmax layer. This architecture can be viewed as a ranker of sentences, and the way to use such a system for summarization is to select a limited amount of sentences from the output according to their probabilities. This is consistent with the earlier work done in the area of extractive text summarization, where most of the models are used as rankers. The schema of our architecture is shown in figure 2.

As we ran our experiments on rather small datasets, we decided not to train our sentence embeddings jointly, but rather use pre-trained embeddings as features. We chose to use Glove embeddings which work well for many NLP tasks, and represent each sentence as a weighted average of the word embeddings based on popularity of each word in the our data.

Results in table 1 show Rouge R1 and R2 numbers for our vanila experiments, where we trained and tested on data coming from the same dataset.

As we can see from the plain vanilla experiments we get very competitive results on the three datasets CNN-DM, DUC and XSum. Although we were unable to create significantly better performing models than the baseline MLE, we will show in the following section that our models perform better in domain transfer experiments by generalizing better. However, it is interesting to notice that RAML is able to get better Rouge R2 value on all the datasets, and better Rouge R1 on the DUC dataset. We were unable to achieve comparable performance with the MRT models on the large CNN-DM dataset due to resources limitation, as MRT training was not fully optimized and took a lot more time to run as we were computing Rouge for all the samples on the fly during training. In the table we also compare our results with a previously published Seq2Seq system published in [Kedzie et al., 2018] and show that we are getting slightly better results.

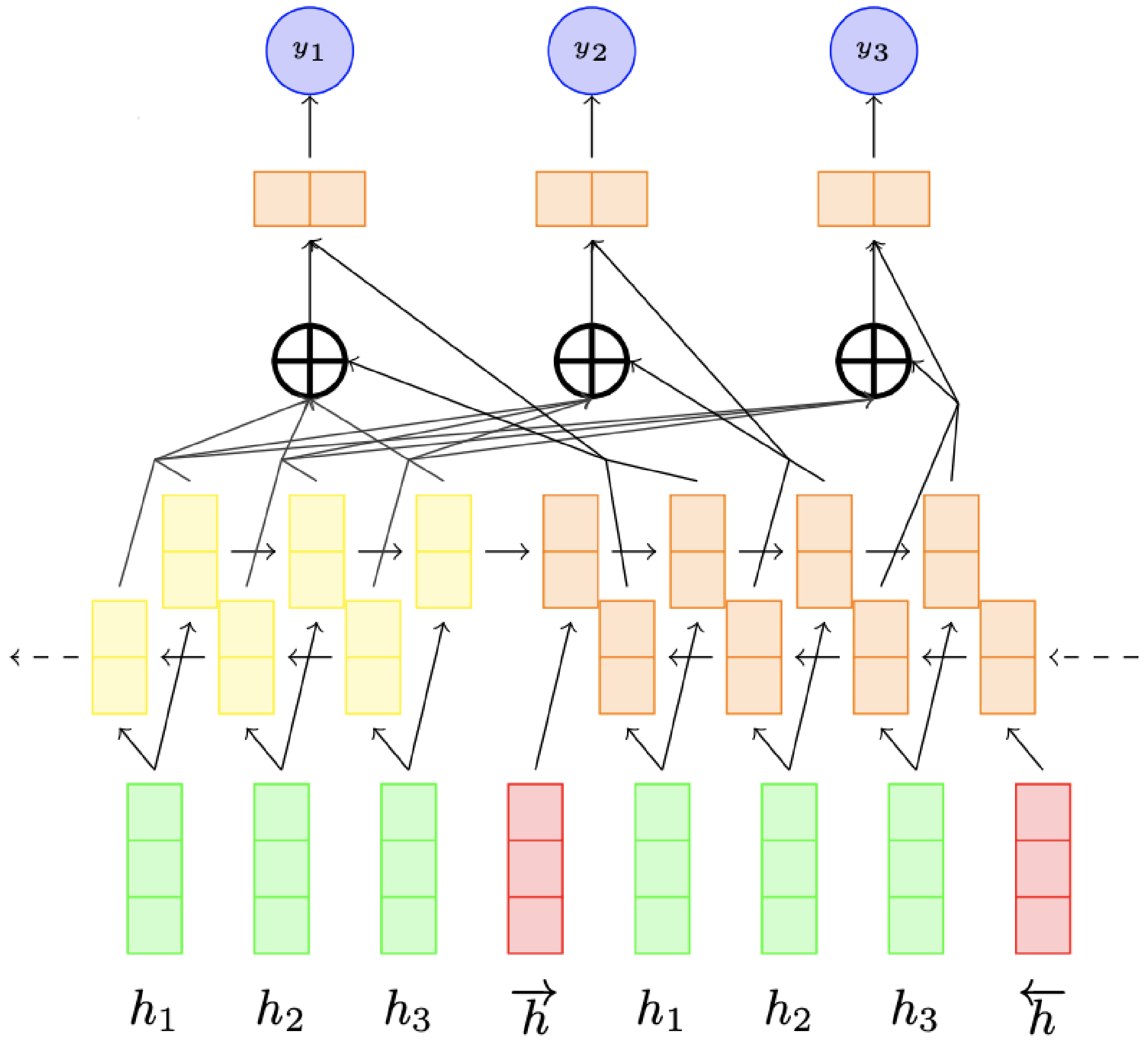


Figure 2: The architecture for the seq-to-seq encoder-decoder models we used throughout our experiments. The sign $\oplus$ stands for Attention.

5 Generalization

In this thesis we claim that poor generalization in machine learning based extractive summarization systems tend to be the result of poor labeling techniques applied to create supervised datasets from human abstracts for each document. To that end we explore several loss functions which we show generalize better than standard MLE approach. We move away from the standard greedy approach for labels extraction described above, and use more sophisticated ones, in which machine learning

XSum	R1	R2
MLE	**47.236**	8.564
RAML	46.924	**8.703**
MRT	46.119	8.581
CNN-DM	R1	R2
MLE	**58.715**	26.727
RAML	58.642	**26.857**
MRT	0.48396	0.17525
Seq2Seq*	–	25.6
DUC-SDS 2002	R1	R2
MLE	49.539	23.927
RAML	**49.771**	**24.011**
MRT	49.482	23.83
Seq2Seq*	–	22.8

Table 1: Performance of our models on different datasets. All evaluations were run with perl Rouge library without excluding stopwords, and with using English porter stemmer. *Seq2Seq is a sequence to sequence deep learning summarization system published in [Kedzie et al., 2018]

algorithms explore a larger space of summaries.

5.1 Related Work

When it comes to using machine learning techniques for extractive summarization, a lot of hidden obstacles uncover themselves and become apparent. One is how to label the needed summaries to be extracted and another is how to evaluate the predictions of a machine learning model. Although it seems like labeling should be an important issue and many researchers should be experimenting with different approaches, not a lot of research has been done in that area so far. Even a very detailed overview of summarization methods in [Dipanjan and Andre, 2007] does not talk much about labeling, focusing more on machine learning architectures and featurization.

It does mention that most approaches use binary labels of whether a sentence should be included or excluded from the summary, and mentions one approach where **soft** labeling was applied. [Svore et al., 2007] trained a machine learning model from the features for each sentence of an article, that could rank the sentences in a test document. The ranking was done using RankNet

([Burges et al., 2005]), a neural network developed to rank a set of inputs that uses gradient descent for training. They used ROUGE-1 to score the similarity between each document in the training set and its human abstract, in a similar fashion as we do in this thesis. This approach of using soft labels aligns well with our work, although we compute the score on the summary level instead of the sentence level, to address the problem of duplication of information in the summary. Also the overview lacks a discussion about how exactly should the **hard** labels be extracted. We think that the way we label the data and thus the kind of loss function we use, determines the ability to generalize well on unseen data.

It is well known that summarization systems tend to not generalize well to new unseen data. That might be due to very different characteristics of the datasets, from which machine learning techniques capture most innate features, and rely solely on these features when presented with unseen data. In [Keneshloo et al., 2018] researchers admit this to be yet an unsolved problem and propose a solution where they use reinforcement learning to create better generalizable models. We also think models which are trained with standard Maximum Likelihood training suffer from inability to generalize well to unseen data. The main difference between their and our approaches are the following:

- They use a specially designed deep learning architecture which allows capturing needed features for generalization, while our methods can be applied to any deep learning architecture with a simple decoder which scores each sentence independently.
- In their approach data from the target distribution has to be present during training, while we do not require target data during training at all.

5.2 Experiments

If a model is trained on the same task but needs to perform well on a different data distribution we call it to be a **domain transfer**. Many times we would like to train a model in a particular environment, but then to be able to reuse the model in another. In the context of summarization,

CNN-DM to XSUM	R1	R2
MLE	46.087	7.893
RAML	**46.527**	**8.034**
MRT	43.912	7.276

CNN-DM to DUC-SDS 2002	R1	R2
MLE	**49.524**	23.851
RAML	49.506	**23.911**
MRT	0.4183	0.15409

Table 2: Results of domain transfer experiments, where we apply models trained on one dataset to unseen data. The upper part describes the results of training on CNN-DM and testing on XSUM, and the bottom part of training on CNN-DM and testing on DUC-SDS 2002

if we train a summarization system on a particular corpus of news for example, we don't want to have to retrain it if we need to summarize letters, webpages, or blogs content. In many cases models trained on one set of data do not perform well on another one. One of the reasons for poor "generalization" performance is that models are trained using Maximum Likelihood approaches and are evaluated on a test set from the same corpus. Such models tend to "overfit" to specific features which are common in the training set, but might not appear in other corpuses.

Our hypothesis was that we could use MRT and RAML procedures to train models on one particular dataset and then transfer the model to another, expecting those models to perform better than their plain vanila Maximum Likelihood counterpart. Our models might benefit from maximizing the actual Rouge score by seeing more positive labels and grasping more of different features, instead of heavily relying on only few important ones.

Results in table 2 show that there is a potential in models trained with RAML, as they seem to perform better on out of domain data. More work is definitely needed to tune hyper-parameter α which controls the Q distribution in MRT and τ which controls the temperature in RAML.

Another type of a text summarization task is called **multi document summarization**, where several documents from multiple sources need to be summarized into one summary. The difficulty compared to the single document summarization task here lies in the fact that there is quite a lot of repetition of content, as the documents for the most part tend to describe the same or similar events.

Method	R1	R2
MLE	32.822	6.55
RAML	33.093	6.328
MRT	29.738	4.718

Table 3: Results on shuffled multi-doc summarization dataset constructed from DUC-MDS

Extractive summarizers have to be careful not to select similar sentences over and over again, as the Rouge metric will not favor these kind of outcomes. We had a hypothesis that MRT and RAML models might perform better in cases where repetition of important content is observed throughout the document. It was partially validated when we created a multi document summarization pipeline from the duc-mds dataset by shuffling all sentences from all the input documents and then running our single document training procedure to train a model. The RAML algorithm show slightly better performance on this task in table 3, partially validating our initial hypothesis.

In order to be able to summarize multiple documents with our models, we had to transform a multi-document dataset into a single document one. This was done by concatenating the content of all the documents in one doc set into one large super document and then shuffling all the sentences. We wanted our models to have equal access to all sentences in all documents during training without giving a bias to a particular document within the doc set. Also, by shuffling the sentences in the document we made the summarization problem more challenging. In our experiments we trained the models on duc-mds 2001 and 2002 and tested on duc-mds 2004

6 Lead Bias

This issue is related to the analysis of what kind of sentences extractive summarization systems tend to return in the summary, after being trained with supervised machine learning techniques. An interesting way to analyze a model in that respect is to examine the distribution of sentences that it is returning when it is used in prediction mode after being trained on a dataset. In many cases when we trained baseline models we have seen that the distribution of sentences is biased to the leading few, meaning the histogram makes it clear there are quite a lot of sentences returned from the head of the document, rather from its other more distant parts. Also, when we compared the Rouge scores of those models with a simple lead-3 baseline (where we select the first 3 sentences to be the summary), we observed that the model is better than the baseline only marginally.

Why does that happen? It seems like it mostly happens in **news** datasets, where important information is served in the beginning of the document, and the details come later. In order to get a summary of such a document, one needs only to select the first few sentences and be done. The modern machine learning techniques, e.g. LSTM-based sequence to sequence models, eagerly learn those position features and rely on them to a large extent. So why is it such a bad idea to just select a few first sentences as the summary?

Apparently not all texts that we want to summarize have this property of bias to the lead. Long scientific articles, blog posts and book chapters are not that susceptible to that illness. However, if we use datasets with documents which are biased to the lead to train our models, they won't generalize well to other domains.

Our hypothesis was that if we use other loss functions and a different way to train models, such that those models are forced in some sense to learn from a richer set of labels, they will be less prone to lead bias problem. In this thesis we explore this direction and show how our models produce distributions with heavier tails, even if without a significant improvement in Rouge.

6.1 Related Work

It is a well known fact that the **lead** baseline for extractive text summarization (according to which a few leading sentences are selected as the summary), especially in the domain of news, achieves relatively good performance in terms of the Rouge metric. In [Brandow et al., 1995] researchers use the lead-3 as the main baseline and in [Wasson, 1998] the search space is limited to the lead portion of the document in commercial search applications to improve precision. They also show that users report high percentages of leads as being an acceptable summary. There are even some examples in more recent research as in [See et al., 2017], where summarization models can't overcome the lead-3 baseline in terms of performance.

That happens mostly in the news domain, where editors put their best content in the head of the document, to make it easier for readers to get the gist of the news quickly and decide whether they want to continue reading. Sometimes it may be used to allure users to read more by giving the most important information first. Training machine learning models on such datasets becomes an extremely difficult task, especially when position features are being used, what happens automatically in sequence based models like RNN's. One approach to combat this phenomena would be to totally discard any positional features, however in experiments with fully shuffled datasets we have seen that the resulting models do not perform well.

6.2 Experiments

First we tried to leverage our wikinews dataset for running MLE training experiments to see how well can we score against the lead-3 on the dataset. In all our experiments we had to perform early stopping as the model capacity was clearly large enough to overfit over the dataset quickly. The following figures 3 and 4 show the training curves of R1 and R2 on the wikinews dataset.

We can see that we were not successful in getting significantly better on the wikinews dataset compared to lead-3. Accordingly, we decided to continue experiments with other already published datasets for text summarization.

One set of experiments we ran in order to focus on the lead-bias problem in summarization, where experiments in which we partitioned the datasets into several portions based on the lead-3 score that each document got with respect to the relevant human abstract. To do that, we scored each document using the lead-3 baseline and sorted the documents ascending by that score. Then we took the first 1/4 of the documents and called it the first quartile, then the next 1/4 of the documents was called the second quartile, etc. We then ran RAML models on these parts of the dataset separately. The basic idea of this exercise was to find evidence that our models work better on non-lead-biased documents. The hypothesis was that MLE model will not do as well as the

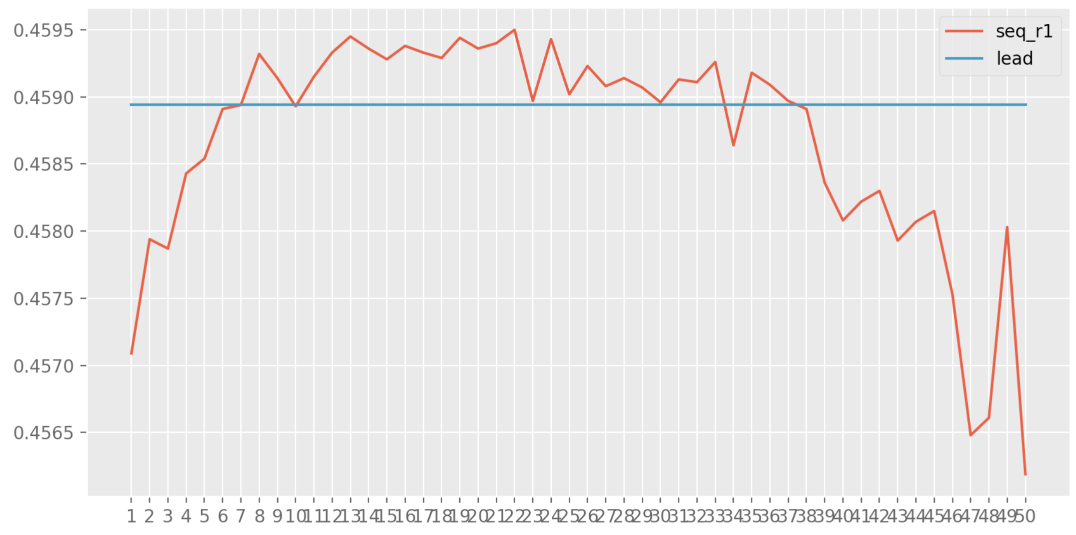


Figure 3: Training standard MLE models on wikinews, R1 curve in red as a function of number of training steps. Lead-3 is depicted in blue.

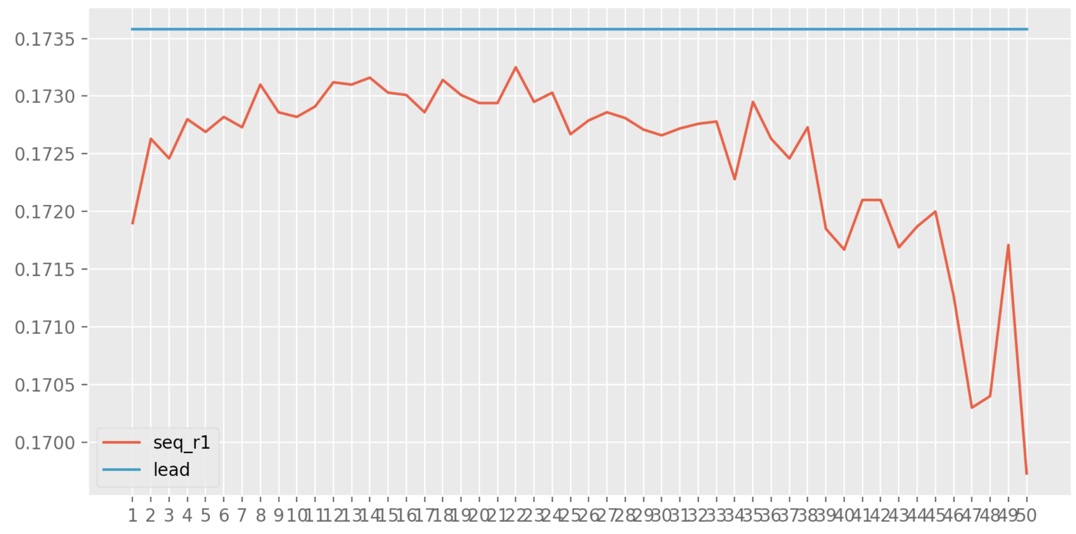


Figure 4: Training standard MLE models on wikinews, R2 curve in red as a function of number of training steps. Lead-3 is depicted in blue.

RAML in the first quartiles - a hypothesis which was validated in our experiments. As we see in table 4, RAML is doing better in the first and the second quartiles, then MLE is getting closer and closing on RAML in the third quartile and getting slightly ahead in the fourth one.

Method	1st quartile R1/R2	2nd quartile R1/R2	3rd quartile R1/R2	4th quartile R1/R2
RAML	44.112 / 14.599	53.807 / 21.715	62.011 / 28.81	74.625 / 42.296
MLE	43.026 / 13.71	53.633 / 21.318	62.491 / 28.831	75.691 / 43.03

Table 4: Results on the lead-3 based partitioned CNN-DM dataset

We also were interested in eyeballing the sentence distribution that different models are producing, as the Rouge metric might not always capture the whole story. To that end we collected a few articles from wikipedia and ran our models on them to investigate what kind of sentences each model tends to return in the summary. We were able to show that different training regimes (like MLE vs. MRT or RAML) tend to generate different distributions of sentences during summary generation. Classical MLE models tend to generate summaries biased towards the first positions of the document, while MRT and RAML tend to generate distributions with heavier tails by selecting more sentences from the middle or the end of the document.

One way to show uniqueness of our models is by examining some of the actual summaries generated on articles from various topics. We wanted to show that our RAML models generate summaries by taking sentences from other places than the head of the document, while the MLE models generate summaries very biased to the lead.

For example, a summary of a wikipedia document about ”capitalism and democracy” from the maximum likelihood model looks like this:

[2]. The extension of universal adult male suffrage in 19th-century britain occurred along with the development of industrial capitalism and democracy became widespread at the same time as capitalism, leading capitalists to posit a causal or mutual relationship between them.

[1]. The relationship between democracy and capitalism is a contentious area in theory and in popular political movements.

[3]. However, according to some authors in the 20th-century capitalism also accompanied a variety of political formations quite distinct from liberal democracies, including fascist regimes, absolute monarchies and single-party states.

[21]. Similarly, suharto's authoritarian reign and extirpation of the communist party of indonesia allowed for the expansion of capitalism in indonesia.

vs. a summary generated using the RAML model:

[1]. The relationship between democracy and capitalism is a contentious area in theory and in popular political movements.

[2]. The extension of universal adult male suffrage in 19th-century britain occurred along with the development of industrial capitalism and democracy became widespread at the same time as capitalism, leading capitalists to posit a causal or mutual relationship between them.

[21]. Similarly, suharto's authoritarian reign and extirpation of the communist party of indonesia allowed for the expansion of capitalism in indonesia.

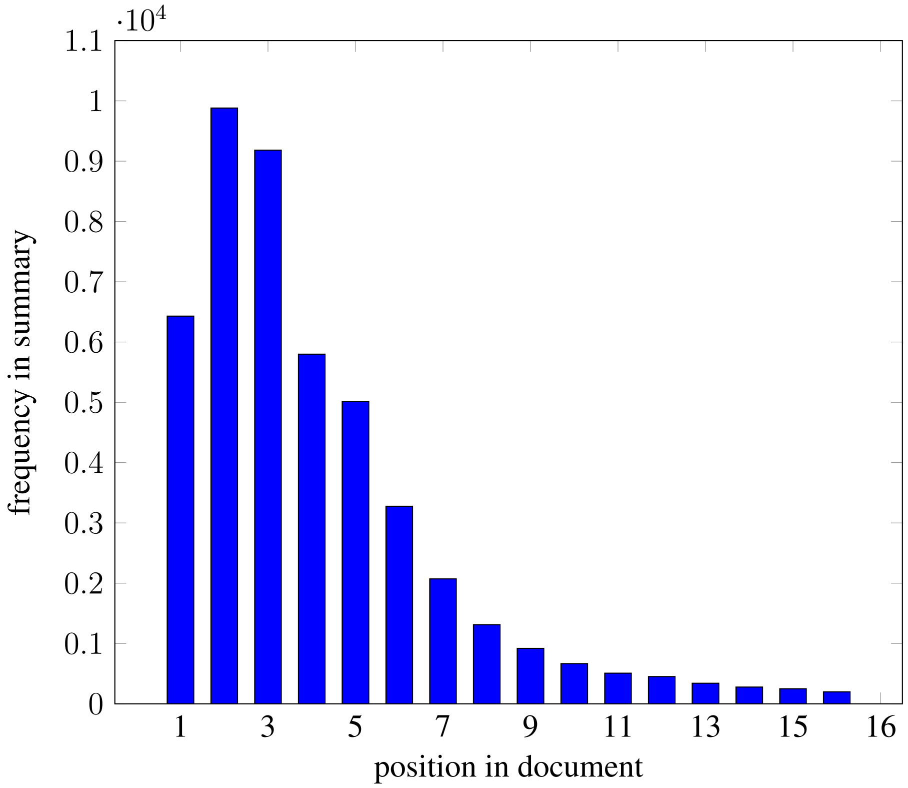
[9]. The ruling on citizens united allows corporations to spend undisclosed and unregulated amounts of money on political campaigns, shifting outcomes to the interests and undermining true democracy.

The following document was fed as input for this explicit summarization experiment:

The relationship between democracy and capitalism is a contentious area in theory and in popular political movements. The extension of universal adult male suffrage in 19th-century Britain occurred along with the development of industrial capitalism and democracy became widespread at the same time as capitalism, leading capitalists to posit a causal or mutual relationship between them. However, according to some authors in the 20th-century capitalism also accompanied a variety of political formations quite distinct from liberal democracies, including fascist regimes, absolute monarchies and single-party states. Democratic peace theory asserts that democracies seldom fight other democracies, but critics of that theory suggest that this may be because of political similarity or stability rather than because they are democratic or capitalist. Moderate critics *argue that though economic growth under capitalism has led to democracy in the past, it may not do so in the future as authoritarian regimes have been able to manage economic growth without making concessions to greater political freedom. One of the biggest supporters of the idea that capitalism promotes political freedom, Milton Friedman, argued that competitive capitalism allows economic and political power to be separate, ensuring that they do not clash with one another. Moderate critics have recently challenged this, stating that the current influence lobbying groups have had on policy in the United States is a contradiction, given the approval of Citizens United. This has led people to question the idea that competitive capitalism promotes political freedom. The ruling on Citizens United allows corporations to spend undisclosed and unregulated amounts of money on political campaigns, shifting outcomes to the interests and undermining true democracy. As explained in Robin Hahnel's writings, the centerpiece of the ideological defense of the free market system is the concept of economic freedom and that supporters equate economic democracy with economic freedom and claim that only the free market system can provide economic freedom. According to Hahnel, there are a few objections to the premise that capitalism offers freedom through economic freedom. These objections are guided by critical questions about who or what decides whose freedoms are more protected. Often, the question of inequality is brought up when discussing how well capitalism promotes democracy. An argument that could stand is that economic growth can lead to inequality given that capital can be acquired at different rates by different people. In Capital in the Twenty-First Century, Thomas Piketty of the Paris School of Economics asserts that inequality is the inevitable consequence of economic growth in a capitalist economy and the resulting concentration of wealth can destabilize democratic societies and undermine the ideals of social justice upon which they are built. Marxists, anarchists (except for anarcho-capitalists) and other leftists argue that capitalism is incompatible with democracy since capitalism according to Marx entails "dictatorship of the bourgeoisie" (owners of the means of production) while democracy entails rule by the people. States with capitalistic economic systems have thrived under political regimes deemed to be authoritarian or oppressive. Singapore has a successful open market economy as a result of its competitive, business-friendly climate and* *robust rule of law. Nonetheless, it often comes under fire for its brand of government which though democratic and consistently one of the least corrupt it also operates largely under a one-party rule and does not vigorously defend freedom of expression given its government-regulated press as well as penchant for upholding laws protecting ethnic and religious harmony, judicial dignity and personal reputation. The private (capitalist) sector in the People's Republic of China has grown exponentially and thrived since its inception, despite having an authoritarian government. Augusto Pinochet's rule in Chile led to economic growth and high levels of inequality by using authoritarian means to create a safe environment for investment and capitalism. Similarly, Suharto's authoritarian reign and extirpation of the Communist Party of Indonesia allowed for the expansion of capitalism in Indonesia.*

This is an interesting example, as the MLE model does select one sentence from the very tail of the document, but it agrees on that with the RAML model. It is the 21-st sentence in the document and it talks about how *the communist party in Indonesia allowed for expansion of capitalism.* However, the models do not agree on all the sentences, and the RAML model selects the 9-th sentence instead of the 3rd as the MLE model does. Basically we can see here that the MLE model selects a few sentences from the head of the document and adds nearly the last sentence, while the RAML model selects the same tail sentence as well, but also includes a sentence from the middle. This pattern repeats itself in many examples in our experiments. Ultimately a pattern which includes one sentence from the head, one closing sentence from the end of the document as a conclusion, and one sentence from the middle, seems like a good one for a summarization system to learn.

Another set of experiments we ran was to analyze the distribution of the sentence positions that each model type is returning in prediction time in a statistical way, by reporting aggregated numbers using a histogram. Table 5 shows the results of those experiments, and figure 5 shows the same results as a histogram. Our hypothesis was that RAML models would have heavier tails than the MLE approach. We were able to verify that hypothesis by showing that indeed our models take more content from the middle and end of the document than the standard MLE models.



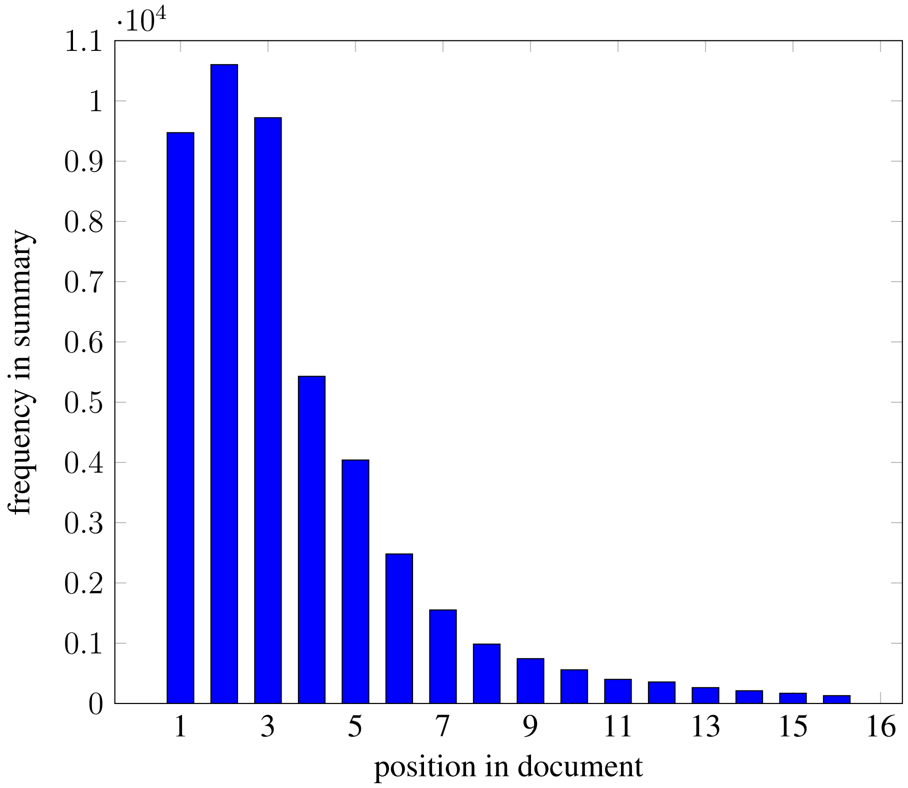


Figure 5: Distribution of sentence positions in summaries run on CNN-DM testset. RAML is on top, and MLE is in the bottom. This is a histogram view of the same data presented in table 5

Position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
RAML	6430	9881	9182	5799	**5014**	**3276**	**2072**	1313	919	667	508	452	341	278	250	199
MLE	**9473**	**10602**	9721	5429	4040	2481	1552	985	744	559	400	357	263	211	169	130

Table 5: Results of sentence distribution experiment on CNN-DM dataset, which shows the distribution of positions the model returns during prediction. The results in the table are limited to first 16 positions just for convenience. We clearly see that RAML is returning more sentences from the middle of the document, while MLE is very lead-biased. For example, we can see that in position 1 and 2 MLE has significantly higher counts of sentences returned, while RAML gets more sentences starting from position 5.

7 Query Focused Summarization

In some scenarios a need arises for a text summary related to a particular topic of interest instead of a general summary of the document, for example, when the document is returned as a result of a search query. In that particular case we don't want the summary solely compress the entire document, but also to fetch a particular portion of the document relevant to the query at hand. General summarization systems fail to be of a good use in such cases, as they lack the ability to focus the outcome on additional information, but rather return generally informative summaries.

7.1 Related Work

There has been quite a bit of recent work on Query Focused text summarization. In [Wang et al., 2016] relevance ordering is produced by a statistical ranker and information coverage with respect to topic distribution, and diverse viewpoints are both encoded as submodular functions, while in [Fetahu et al., 2013] researchers automatically generate structured summaries from user queries that uses POS patterns. Bayesian approach has been applied to the problem of query focused summarization in [Hal, 2009].

Another somewhat related field which flourished with the advancement of deep learning architectures is the field of Question Answering. The task there is to find an answer to a question given a document as input text. This document sometimes may not contain the answer, and then the model is expected to indicate that the answer cannot be found in the given text. This seems to be a more complex problem than just trying always to find some portion of text which suits to be the answer to the question, and if in reality no such answer exists in the passage, returning some portion of the text, without really being penalized for returning unrelated text.

It seems like the same particular advancements in deep learning which made it possible for neural translation systems to become state of the art in the field of machine translation, also made it possible for the neural question answering systems to start leading in their field. The main novel idea which made it possible was **Attention**, a mechanism which first was introduced for neural machine translation in [Bahdanau et al., 2014] where the network is capable to align original and translated text. Since then, the mechanism of attention has been used in many contexts and even was used as a basic building block for a new neural architecture called **Transformer** introduced in [Seo et al., 2016]. Since then researchers started to use attention for other tasks like Question Answering, as in [Seo et al., 2016], where an attention flow architecture is using several types of attention to match pieces of text with the question.

It really seems like the 2015 and 2016 years were very fruitful for QA, as a slew of research has been done in that area then. [Andreas et al., 2016] describe a very interesting and unique question answering model. That system is applicable to structured knowledge bases. The model is using question and answers as supervision to learn how to compose existing neural modules for solving the problem at hand. This approach departs significantly in technique and style from other recent end-to-end deep learning systems like in [Seo et al., 2016]. Another interesting research from [Weston et al., 2015] is focusing on how to evaluate question answering models by preparing a set of tests that an intelligent system has to perform well on in order to prove that it really possesses ”understanding” capabilities, e.g. ability to answer questions via chaining facts, perform simple induction, deduction and etc.

7.2 Experiments with Squad Dataset

We wanted to see if we can apply machine learning techniques to the problem of query focused extractive summarization. The main problem is the lack of good labeled data, which contains both the search query and the summary. To that end we decided to repurpose the Squad data, originally created for Question Answering task.

SQUAD stands for Stanford Question Answering Dataset, a recently published dataset for reading comprehension and question answering([Rajpurkar et al., 2016]). It consists of questions asked by crowd-workers on Wikipedia articles and answers in the form of a text segment from a reading passage. SQuAD2.0 combines the original 100K questions in SQuAD1.1 with more than 50K new questions which do not have answers, designed by the crowd-workers to be very similar to the answerable ones. Thus to perform well on the second dataset, systems have to be robust enough to be able to detect an unanswerable question.

Originally the Squad dataset was created for the Question Answering task, where a question and a document are presented to the system as input, and the system's task is to find the short answer to the question in the document. We have adapted this dataset to be a suitable dataset for query focused summarization task, in which there is an additional piece of information given to the summarization system with each document in a form of text query. The intention of the query is to make it possible for the summarization system to focus and return a more suitable summary. This works well in the context of a search engine for example, where not just a general summary of a document is needed, but rather a summary focused on the original search query the user provided.

The process of adaptation of the Squad dataset for the extractive query focused summarization task was the following:

- We concatenated all the passages into one large piece of text and called that the document
- Each question was perceived to be one search query
- The passage containing the answer to the question was defined to be the summary

We used that transformed dataset to train a deep learning summarization system by maximizing the cross entropy with respect to the correct labels of sentences to include in the summary. When trying to optimize the maximum likelihood objective on this dataset, we notices that the loss is not going down significantly and the resulting performance in terms of Rouge was not as expected. Although we were not able to actually train good performing models on this dataset, we think this approach is still promising and needs more research. One potential issue which might have caused too much variance during training, was varying location of the answer to the question inside the passage, and so the model could not capture the similarity features between the passage and the query. One way to remedy that issue is to extract a passage which surrounds the answer with one sentence before and one sentence after, so that the context for the question in the passage is always of constant length and the location of the answer is always in the middle. We think this might be a good future research idea.

7.3 QA for Query Focused Summarization

We experimented with another adaptation as we had a hypothesis that a good performing Query Focused summarization system can be implemented using an existing Question Answering model, where the **query** is transformed into a **question**, and the answer from the QA system is turned into a summary.

We have built such a system based on a deep QA model developed by the Allennlp group, described in details in [Seo et al., 2016]. The model uses a special Attention Flow mechanism which allows it to accurately select the start and the end of the most likely portion of text to contain the answer to the question. Throughout running the experiments we noticed that the model is very sensitive to the presence of the question word in the question text, presumably because in the training data each question snippet contained a question word, so the learning algorithm was able to extract these question words as strong features. It does make sense as the question word in many cases describes the type of the answer to look for: for ”where” look for locations, for ”when” look for entities describing time, etc. As we wanted to develop a generic system, we could not use one particular question word for all our queries, but instead decided to implement a rankings merging mechanism, which merges all the outputs from all the different questions.

A $query$ was transformed into a question using a question word $qword$ in the following way:

$$Question(query, qword) = qword \oplus \text{"}is\text{"} \oplus query \oplus \text{"}?\text{"} \tag{28}$$

where $\oplus$ stands for "concatenation", so for example a query "table" was transformed into a "Where is table?" question, which was fed into the QA system afterwords.

The output of the QA system was defined by the following:

$$QA(Question(q), D) = \Bigg\{ P(start(i)), P(end(i)) \Bigg\}_{i \in D} \tag{29}$$

where q is the query, D is the document, $start(i)$ and $end(i)$ represented the events of i being the token of start and end of the answer portion respectfully.

We then ranked each sentence s_k in the document by the likelihood of that sentence to contain the answer $P(s_k)$:

$$P(s_k) = \max_{i,j \in s_k} \Bigg\{ P(start(i)) \times P(end(j)) \Bigg\} \tag{30}$$

After applying the QA system on the same document but with different question words (we used "what", "when", "where", "why", "how"), we got 5 different rankings of the same sentences. To merge all these rankings we used the **Borda count** which gives a score to each object equal to the number of objects it has outranked. So if the ranking is "1","5","10","7", then object "5" has a Borda count of 2, and "10" has a Borda count of 1, etc. Once we get scores for each ranked object we can add those scores and rank the objects according to the total scores to get a final ranking.

To create a summary out of ranked documents, we just take the first k according to a budget constraint imposed beforehand (in our experiments we use 100 words). We have run several experiments where humans had to judge the quality of the summaries given the query by indicating whether the whole document was relevant to the query just based on our summary excerpt. In those experiments we always knew which documents really were relevant and which were not, so we were able to assess the precision and recall of the summarization technique as a classifier of relevance of the document given the query. In the experiments which involved the aforementioned technique we saw precision and recall going up due to the inclusion of the QA-system based rankings, along with some other baselines, which included simple word matchers and different similarity based matchers as well.

8 Conclusions and Future Research

This thesis addresses a few interesting problems in the realm of extractive text summarization. We introduce an adaptation of Minimum Risk Training and Reward Augmented Maximum Likelihood for the extractive text summarization task, and pinpoint the advantages these methods of training have over the standard Maximum Likelihood Estimation. Generalization and Lead Bias in particular presented as the two problems which can benefit from our approach with some experimental results shown as evidence to that end. Furthermore, we talk about Query Focused Extractive Text Summarization and the ways one can reuse the Question Answering datasets and models to tackle this problem in an unconventional way.

Training models which sample from a dynamic distribution like in MRT is quite difficult, as the variance of the estimate of the gradient gets very large. Special tricks and techniques have to be applied in order to make this kind of an approach really work in practice. We have affirmed through experiments that the same problems which prevail in the reinforcement learning literature with respect to the difficulties of direct optimization of gradient policy methods, exist in MRT and a significant amount of hyper-parameters tuning with the Q distribution hyper parameter α is needed to get the training working well. Another conclusion we have arrived at is that the community needs a good implementation of Rouge in python in order to be able to incorporate it into training procedures like MRT and RAML, which either needs to run this computation online for each sample, or over large amount of candidates offline.

However we have seen that training deep models with loss functions which explore the space of candidates more than the standard Maximum Likelihood Estimation, improve performance, if not for all documents, at least for those which are not lead biased to the human abstract. We think that an interesting path for future research might lie in the area of when to apply models like RAML vs. regular MLE models. In our experiments, if we condition on the lead-3 score of a document with respect to the human abstract, and chose a model appropriately, we got far better performance than each of those models gets on its own. This can be seen as building an ensemble method with a smart conditioning for invocation of each model on different documents. We think more research is needed to create efficient and good performing ensembles of models for extractive text summarization which leverage our findings.

On the query focused summarization front we think that using QA models and adapting Question Answering datasets for the focused summarization task has great potential. There is clearly a relation between these two tasks and more work is needed to establish even a stronger link. For example, in our experiments we used several different question words to created a question out of a query. Future research is needed to investigate a possibility of learning the appropriate question word or even the whole question given the query and the document. May be machine translation models can be applied to translate a query into a question given the document and then used in a QA model to generate query focused summaries.

References

[Andreas et al., 2016] Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Learning to compose neural networks for question answering. *CoRR*, abs/1601.01705.

[Arora et al., 2017] Arora, S., Liang, Y., and Ma, T. (2017). A simple but tough-to-beat baseline for sentence embeddings. In *ICLR*. OpenReview.net.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Berg-Kirkpatrick et al., 2011] Berg-Kirkpatrick, T., Gillick, D., and Klein, D. (2011). Jointly learning to extract and compress. In Lin, D., Matsumoto, Y., and Mihalcea, R., editors, *ACL*, pages 481–490. The Association for Computer Linguistics.

[Brandow et al., 1995] Brandow, R., Mitze, K., and Rau, L. (1995). Automatic condensation of electronic publications by sentence selection. 31:675–685.

[Burges et al., 2005] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96, New York, NY, USA. ACM Press.

[Conroy and O'Leary, 2001] Conroy, J. M. and O'Leary, D. P. (2001). Text summarization via hidden markov models. In *Research and Development in Information Retrieval*, pages 406–407.

[Degris et al., 2012] Degris, T., Pilarski, P. M., and Sutton, R. S. (2012). Model-free reinforcement learning with continuous action in practice. In *ACC*, pages 2177–2182. IEEE.

[Dipanjan and Andre, 2007] Dipanjan, D. and Andre, F.T., M. (2007). A survey on automatic text summarization.

[Fetahu et al., 2013] Fetahu, B., Pereira Nunes, B., and Dietze, S. (2013). Towards focused knowledge extraction: Query-based extraction of structured summaries. In *In Proceedings of the 22nd World Wide Web Conference*. ACM.

[Hal, 2009] Hal, D. (2009). Bayesian query-focused summarization. *CoRR*, abs/0907.1814.

[He and Deng, 2012] He, X. and Deng, L. (2012). Maximum expected bleu training of phrase and lexicon translation models. In *ACL (1)*, pages 292–301. The Association for Computer Linguistics.

[Hermann et al., 2015] Hermann, K. M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching machines to read and comprehend. cite arxiv:1506.03340Comment: Appears in: Advances in Neural Information Processing Systems 28 (NIPS 2015). 14 pages, 13 figures.

[Jiao et al., 2018] Jiao, X., Wang, F., and Feng, D. (2018). Convolutional neural network for universal sentence embeddings. In Bender, E. M., Derczynski, L., and Isabelle, P., editors, *COLING*, pages 2470–2481. Association for Computational Linguistics.

[Kedzie et al., 2018] Kedzie, C., McKeown, K., and III, H. D. (2018). Content selection in deep learning models of summarization. *CoRR*, abs/1810.12343.

[Keneshloo et al., 2018] Keneshloo, Y., Ramakrishnan, N., and Reddy, C. K. (2018). Deep transfer reinforcement learning for text summarization. *CoRR*, abs/1810.06667.

[Koehn et al., 2003] Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

[Kupiec et al., 1995] Kupiec, J., Pedersen, J., and Chen, F. (1995). A trainable document summarizer.

[Lin, 2004] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Proc. ACL workshop on Text Summarization Branches Out*, page 10.

[Miles, 2002] Miles, O. (2002). Using maximum entropy for sentence extraction. In *Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4*, pages 1–8.

[Narayan et al., 2018a] Narayan, S., Cohen, S. B., and Lapata, M. (2018a). Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *CoRR*, abs/1808.08745.

[Narayan et al., 2018b] Narayan, S., Cohen, S. B., and Lapata, M. (2018b). Ranking sentences for extractive summarization with reinforcement learning. In Walker, M. A., Ji, H., and Stent, A., editors, *NAACL-HLT*, pages 1747–1759. Association for Computational Linguistics.

[Norouzi et al., 2016] Norouzi, M., Bengio, S., Chen, Z., Jaitly, N., Schuster, M., Wu, Y., and Schuurmans, D. (2016). Reward augmented maximum likelihood for neural structured prediction. *CoRR*, abs/1609.00150.

[Rajpurkar et al., 2016] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

[Ranzato et al., 2016] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. In Bengio, Y. and LeCun, Y., editors, *ICLR*.

[See et al., 2017] See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368.

[Seo et al., 2016] Seo, M. J., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

[Shen et al., 2007] Shen, D., Sun, J.-T., Li, H., Yang, Q., and Chen, Z. (2007). Document summarization using conditional random fields. In Veloso, M. M., editor, *IJCAI*, pages 2862–2867.

[Shen et al., 2015] Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2015). Minimum risk training for neural machine translation. *CoRR*, abs/1512.02433.

[Smith and Eisner, 2006] Smith, D. A. and Eisner, J. (2006). Minimum risk annealing for training log-linear models. In Calzolari, N., Cardie, C., and Isabelle, P., editors, *ACL*. The Association for Computer Linguistics.

[Sutton et al., 2000] Sutton, R., McAllester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

[Sutton and Andrew, 2017] Sutton, Richard, R. and Andrew, Barto, G. (2017). Reinforcement learning: An introduction. In *The MIT Press*.

[Svore et al., 2007] Svore, K. M., Vanderwende, L., and Burges, C. J. C. (2007). Enhancing single-document summarization by combining ranknet and third-party sources. In Eisner, J., editor, *EMNLP-CoNLL*, pages 448–457. ACL.

[Tsutomu et al., 2002] Tsutomu, H., Yutaka, S., Hideki, I., and Eisaku, M. (2002). Ntts text summarization system for duc-2002. In *Citeseer*.

[Wang et al., 2016] Wang, L., Raghavan, H., Cardie, C., and Castelli, V. (2016). Query-focused opinion summarization for user-generated content. *CoRR*, abs/1606.05702.

[Wasson, 1998] Wasson, M. (1998). Using leading text for news summaries: Evaluation results and implications for commercial summarization applications. In Boitet, C. and Whitelock, P., editors, *COLING-ACL*, pages 1364–1368. Morgan Kaufmann Publishers / ACL.

[Weston et al., 2015] Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., and Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. cite arxiv:1502.05698.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256.

[Yao et al., 2018] Yao, K., Zhang, L., Luo, T., and Wu, Y. (2018). Deep reinforcement learning for extractive document summarization. *Neurocomputing*, 284:52–62.